

corresponding to US 6,317,115 B1 and US 6,693,635 B1

(19) 日本国特許庁 (J P)

(12) 公開特許公報 (A)

(11) 特許出願公開番号

特開平8-138033

(43) 公開日 平成8年(1996)5月31日

(51) Int.Cl. <sup>8</sup>	識別記号	序内整理番号	F I	技術表示箇所
G 0 6 T 1/00				
G 0 6 F 13/00	3 5 7 Z	7368-5E 9365-5H	G 0 6 F 15/ 62	

審査請求 未請求 請求項の数34 O L (全 22 頁)

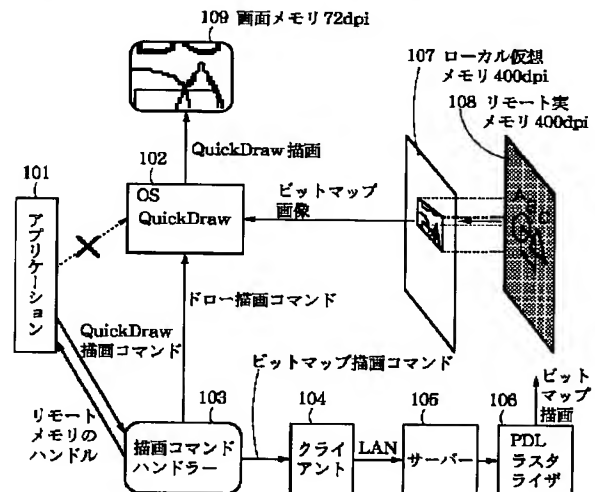
(21) 出願番号	特願平6-280622	(71) 出願人	000001007 キヤノン株式会社 東京都大田区下丸子3丁目30番2号
(22) 出願日	平成6年(1994)11月15日	(72) 発明者	横溝 良和 東京都大田区下丸子3丁目30番2号キヤノ ン株式会社内
		(74) 代理人	弁理士 丸島 健一

(54) 【発明の名称】 画像編集装置とその方法及び画像表示制御装置とその方法及び画像編集装置と画像表示制御装置から構成されるシステム

(57) 【要約】 (修正有)

【目的】 クライアントの画像描画処理をサーバーに代行させることができる。

【構成】 アプリケーション101からオペレーティングシステム102に対して発行される描画関数コールを捉えてサーバー105に前記描画関数コールに対応するスクリプトを発行し、該スクリプトを解釈して前記サーバー105上に確保されたリモート実メモリ108上に画像を描画し、該描画された描画画像データをネットワークを介して前記リモート実メモリ108から前記オペレーティングシステムに取り込み、画面メモリ109に表示する構成を特徴とする。



## 【特許請求の範囲】

【請求項1】 画像処理を実行するサーバーと接続された画像表示制御装置であって、前記サーバーに描画命令を出力する出力手段と、前記出力手段で出力した前記描画命令を解釈して、前記サーバー上に確保された画像描画メモリ上に描画された画像データを取得する取得手段と、前記取得手段で取得した前記画像データを画面メモリに表示する表示制御手段とを有することを特徴とする画像表示制御装置。

【請求項2】 前記サーバー上に確保された前記画像描画メモリは、前記画像表示制御装置の前記画面メモリより大きいことを特徴とする請求項1記載の画像表示制御装置。

【請求項3】 前記サーバー上に確保された前記画像描画メモリの解像度は、前記画像表示制御装置の前記画面メモリの解像度より高いことを特徴とする請求項1記載の画像表示制御装置。

【請求項4】 前記画像表示制御装置の画面メモリに表示される画像データは、前記サーバー上に確保された前記画像描画メモリに描画される画像データの一部であることを特徴とする請求項1記載の画像表示制御装置。

【請求項5】 前記画像表示制御装置の画面メモリに表示される画像データをスクロールすると、前記サーバー上に確保された前記画像描画メモリに描画される画像データの一部が前記画面メモリに複写されることを特徴とする請求項1記載の画像表示制御装置。

【請求項6】 前記画像表示制御装置は、ホストコンピュータであることを特徴とする請求項1記載の画像表示制御装置。

【請求項7】 前記サーバーと前記画像表示制御装置は、ネットワークを介して接続されていることを特徴とする請求項1記載の画像表示制御装置。

【請求項8】 ホストコンピュータと接続された画像編集装置であって、前記ホストコンピュータから描画命令を入力する入力手段と、前記入力手段で入力した前記描画命令を解釈して、確保した画像描画メモリ上に画像データを描画する描画手段と、前記描画手段で描画した前記画像データを前記ホストコンピュータの画面メモリに表示する為に、前記画像データを前記ホストコンピュータへ出力する出力手段とを有することを特徴とする画像編集装置。

【請求項9】 前記画像編集装置に確保された前記画像描画メモリは、前記ホストコンピュータの前記画面メモリより大きいことを特徴とする請求項8記載の画像編集装置。

【請求項10】 前記画像編集装置に確保された前記画像描画メモリの解像度は、前記ホストコンピュータの前

記画面メモリの解像度より高いことを特徴とする請求項8記載の画像編集装置。

【請求項11】 前記ホストコンピュータの画面メモリに表示される画像データは、前記画像編集装置に確保された前記画像描画メモリに描画される画像データの一部であることを特徴とする請求項8記載の画像編集装置。

【請求項12】 前記ホストコンピュータの画面メモリに表示される画像データをスクロールすると、前記画像編集装置に確保された前記画像描画メモリに描画される画像データの一部が前記画面メモリに複写されることを特徴とする請求項8記載の画像編集装置。

【請求項13】 前記画像編集装置は、サーバーであることを特徴とする請求項8記載の画像編集装置。

【請求項14】 前記ホストコンピュータと前記画像編集装置は、ネットワークを介して接続されていることを特徴とする請求項8記載の画像編集装置。

【請求項15】 画像処理を実行するサーバーとホストコンピュータとがネットワークに接続されたシステムであって、

20 前記ホストコンピュータは、前記サーバーに描画命令を出力する出力手段と、前記出力手段で出力した前記描画命令を解釈して、前記サーバー上に確保された画像描画メモリ上に描画された画像データを取得する取得手段と、前記取得手段で取得した前記画像データを画面メモリに表示する表示制御手段とを有し、前記サーバーは、前記ホストコンピュータから描画命令を入力する入力手段と、  
30 前記入力手段で入力した前記描画命令を解釈して、確保した画像描画メモリ上に画像データを描画する描画手段と、前記描画手段で描画した前記画像データを前記ホストコンピュータの画面メモリに表示する為に、前記画像データを前記ホストコンピュータへ転送する転送手段とを有することを特徴とする。

40 【請求項16】 前記サーバー上に確保された前記画像描画メモリは、前記ホストコンピュータの前記画面メモリより大きいことを特徴とする請求項15記載のシステム。

【請求項17】 前記サーバー上に確保された前記画像描画メモリの解像度は、前記ホストコンピュータの前記画面メモリの解像度より高いことを特徴とする請求項15記載のシステム。

【請求項18】 前記ホストコンピュータの画面メモリに表示される画像データは、前記サーバー上に確保された前記画像描画メモリに描画される画像データの一部であることを特徴とする請求項15記載のシステム。

50 【請求項19】 前記ホストコンピュータの画面メモリに表示される画像データをスクロールすると、前記サー

バー上に確保された前記画像描画メモリに描画される画像データの一部分が前記画面メモリに複写されることを特徴とする請求項15記載のシステム。

【請求項20】 前記サーバーと複数のホストコンピュータが前記ネットワークに接続されていることを特徴とする請求項15記載のシステム。

【請求項21】 画像処理を実行するサーバーと接続された画像表示制御装置における画像表示制御方法であって、  
前記サーバに描画命令を出力する出力工程と、  
前記出力工程で出力した前記描画命令を解釈して、前記サーバー上に確保された画像描画メモリ上に描画された画像データを取得する取得工程と、  
前記取得工程で取得した前記画像データを画面メモリに表示する表示制御工程とを有することを特徴とする画像表示制御方法。

【請求項22】 前記サーバー上に確保された前記画像描画メモリは、前記画像表示制御装置の前記画面メモリより大きいことを特徴とする請求項21記載の画像表示制御方法。

【請求項23】 前記サーバー上に確保された前記画像描画メモリの解像度は、前記画像表示制御装置の前記画面メモリの解像度より高いことを特徴とする請求項21記載の画像表示制御方法。

【請求項24】 前記画像表示制御装置の画面メモリに表示される画像データは、前記サーバー上に確保された前記画像描画メモリに描画される画像データの一部分であることを特徴とする請求項21記載の画像表示制御方法。

【請求項25】 前記画像表示制御装置の画面メモリに表示される画像データをスクロールすると、前記サーバー上に確保された前記画像描画メモリに描画される画像データの一部分が前記画面メモリに複写されることを特徴とする請求項21記載の画像表示制御方法。

【請求項26】 前記画像表示制御装置は、ホストコンピュータであることを特徴とする請求項21記載の画像表示制御方法。

【請求項27】 前記サーバーと前記画像表示制御装置は、ネットワークを介して接続されていることを特徴とする請求項21記載の画像表示制御方法。

【請求項28】 ホストコンピュータと接続された画像編集装置における画像編集方法であって、  
前記ホストコンピュータから描画命令を入力する入力工程と、  
前記入力工程で入力した前記描画命令を解釈して、確保した画像描画メモリ上に画像データを描画する描画工程と、  
前記描画工程で描画した前記画像データを前記ホストコンピュータの画面メモリに表示する為に、前記画像データを前記ホストコンピュータへ出力する出力工程とを有

することを特徴とする画像編集方法。

【請求項29】 前記画像編集装置に確保された前記画像描画メモリは、前記ホストコンピュータの前記画面メモリより大きいことを特徴とする請求項28記載の画像編集方法。

【請求項30】 前記画像編集装置に確保された前記画像描画メモリの解像度は、前記ホストコンピュータの前記画面メモリの解像度より高いことを特徴とする請求項28記載の画像編集方法。

10 【請求項31】 前記ホストコンピュータの画面メモリに表示される画像データは、前記画像編集装置に確保された前記画像描画メモリに描画される画像データの一部分であることを特徴とする請求項28記載の画像編集方法。

【請求項32】 前記ホストコンピュータの画面メモリに表示される画像データをスクロールすると、前記画像編集装置に確保された前記画像描画メモリに描画される画像データの一部分が前記画面メモリに複写されることを特徴とする請求項28記載の画像編集方法。

20 【請求項33】 前記画像編集装置は、サーバーであることを特徴とする請求項28記載の画像編集方法。

【請求項34】 前記ホストコンピュータと前記画像編集装置は、ネットワークを介して接続されていることを特徴とする請求項28記載の画像編集方法。

【発明の詳細な説明】

【0001】

【産業上の利用分野】本発明は、ネットワークに接続されたホストコンピュータ等の画像表示制御装置及びその方法とサーバー等の画像編集装置及びその方法と当該画像表示制御装置と当該画像編集装置から構成されるシステムに関する。

【0002】

【従来の技術】従来、ネットワーク化の進展に伴い種々の端末が通信機能を持つようになると、相互接続性が非常に重要になってくる。例えば、Faxの場合、公衆網にアクセスする方法に関してはC C I T Tが規定したファクシミリ通信手順が存在するが、Faxサーバーとしてネットワーク化しようとする場合の手順は標準化されていない。

40 【0003】また、リレーショナルデータベース(RDB)に関しては、一応ANSIのSQL言語が標準としての地位を確立しているが、RDBへのアクセスプロトコルやフロントエンドのインタフェース(API)は各社バラバラである。印刷に関しては、OS毎の業界標準は存在するが、マルチベンダー環境におけるネットワークプリンタのアーキテクチャは標準化されていない。

【0004】OCRに関しては、従来ネットワーク化の概念がなかったために、ネットワークOCRのアクセス方法はこれから決まってくる。ネットワーク上の画像処理アクセラレータの概念も従来はなかった。このよう

に、データベースならデータベース、ネットワークプリンタならネットワークプリンタといった個別アプリケーション分野毎の標準化の努力はなされているが、分野をまたがる標準化はない。例えば、Faxサーバーで受信した画像をOCRサーバーでコードに変換し、それをデータベースに蓄積するといった一連の作業を行うには、各サーバー間の連携はないから、クライアント側でFaxとOCRとデータベースの3本のクライアント側プログラムを起動し順次作業を進めていくことになる。それでも、もし3本のプログラムが、最新のOSの提供する機能をフルに用いて、アプリケーション間通信を行えば、上記の作業は自動化できるかも知れない。しかし、最新のOSはアプリケーション間通信の仕組みは提供しているが、そのやり方はベンダー任せであり、何も決まっていない。

#### 【0005】

【発明が解決しようとする課題】しかしながら、従来のサーバーシステムにおいては、例えば、Faxサーバーで受信した画像をOCRサーバーでコードに変換し、それをデータベースに蓄積するといった一連の作業を行うためには、各サーバー間の連携はないから、クライアント側でFaxフロントエンドとOCRフロントエンドとデータベースフロントエンドの3本のフロントエンドプログラムを起動し順次作業を進めていくことになる。しかし、その作業は多くの「カット＆ペースト」や「一時ファイル」へのアクセスの繰り返しとなり苦痛を伴う。

【0006】また、個々のサーバーへのアクセス方法は、サーバーが発明された歴史上の必然によりバラバラで、サービスの分野にまたがるアクセスはほとんど不可能である。本発明は、ネットワーク上に分散している独立サーバーを仮想的に統合し、巨大な仮想サーバーを論理的に構築することにより、サーバー毎の互換性をなくし、1本のアプリケーションから簡単に多くのサーバー機能を利用可能ならしむためのものである。

【0007】また、従来、小さなパーソナルコンピュータではメモリの不足により、大画面、高精細画像の編集はできなかった。例えば、400dpi（ドット/インチ）でA4サイズのフルカラー画像は、47メガバイトものメモリを必要とする。パーソナルコンピュータでこの様に大きな画像を扱える様にするのはコスト的にも、速度的にも、非現実的であり、信頼性の面からも問題であった。なぜならシングルチップアーキテクチャの制約から、パーソナルコンピュータのCPUパワーは、それほど大きくなく、速度的に遅くなる。また、ハードディスクをRAMの一部に取込む方法だと、スワップが発生する度に絶望的な遅延が発生する。また、今日のパーソナルコンピュータのOSの場合、この様に大きなメモリを扱っていると、クラッシュしてしまう事が多い。

【0008】かかる問題点を解決するために本発明の目的は、サーバに描画命令を出力し、出力した前記描画命

令を解釈してサーバー上に確保された画像描画メモリ上に描画された画像データを取得し、取得した画像データを画面メモリに表示する画像表示制御装置及びその方法を提供することである。

【0009】かかる問題点を解決するために本発明の目的は、ホストコンピュータから描画命令を入力し、入力した前記描画命令を解釈して、確保した画像描画メモリ上に画像データを描画し、描画した画像データをホストコンピュータの画面メモリに表示する為に、画像データをホストコンピュータへ出力する画像編集装置及びその方法を提供することである。

【0010】かかる問題点を解決するために本発明の目的は、サーバに描画命令を出力し、出力した前記描画命令を解釈して、サーバー上に確保された画像描画メモリ上に描画された画像データを取得し、取得した前記画像データを画面メモリに表示する表示制御手段とを有するホストコンピュータと、ホストコンピュータから描画命令を入力し、入力した前記描画命令を解釈して、確保した画像描画メモリ上に画像データを描画し、描画した前記画像データを前記ホストコンピュータの画面メモリに表示する為に、画像データをホストコンピュータへ転送するサーバーとから構成されるシステムを提供することである。

#### 【0011】

【課題を解決するための手段】上記問題点を解決するために本発明の画像表示制御装置は、画像処理を実行するサーバーと接続された画像表示制御装置であって、前記サーバに描画命令を出力する出力手段と、前記出力手段で出力した前記描画命令を解釈して、前記サーバ上に確保された画像描画メモリ上に描画された画像データを取得する取得手段と、前記取得手段で取得した前記画像データを画面メモリに表示する表示制御手段とを有する。

【0012】上記問題点を解決するために本発明の画像編集装置は、ホストコンピュータと接続された画像編集装置であって、前記ホストコンピュータから描画命令を入力する入力手段と、前記入力手段で入力した前記描画命令を解釈して、確保した画像描画メモリ上に画像データを描画する描画手段と、前記描画手段で描画した前記画像データを前記ホストコンピュータの画面メモリに表示する為に、前記画像データを前記ホストコンピュータへ出力する出力手段とを有する。

【0013】上記問題点を解決するために本発明のシステムは、画像処理を実行するサーバーとホストコンピュータとがネットワークに接続されたシステムであって、前記ホストコンピュータは、前記サーバに描画命令を出力する出力手段と、前記出力手段で出力した前記描画命令を解釈して、前記サーバ上に確保された画像描画メモリ上に描画された画像データを取得する取得手段と、前記取得手段で取得した前記画像データを画面メモリに

表示する表示制御手段とを有し、前記サーバーは、前記  
 ホストコンピュータから描画命令を入力する入力手段  
 と、前記入力手段で入力した前記描画命令を解釈して、  
 確保した画像描画メモリ上に画像データを描画する描画  
 手段と、前記描画手段で描画した前記画像データを前記  
 ホストコンピュータの画面メモリに表示する為に、前記  
 画像データを前記ホストコンピュータへ転送する転送手  
 段とを有する。

【0014】上記問題点を解決するために本発明の画像  
 表示制御方法は、画像処理を実行するサーバーと接続さ  
 れた画像表示制御装置における画像表示制御方法であ  
 って、前記サーバに描画命令を出力する出力工程と、前記  
 出力工程で出力した前記描画命令を解釈して、前記サー  
 バー上に確保された画像描画メモリ上に描画された画像  
 データを取得する取得工程と、前記取得工程で取得した  
 前記画像データを画面メモリに表示する表示制御工程と  
 を有する。

【0015】上記問題点を解決するために本発明の画像  
 編集方法は、ホストコンピュータと接続された画像編集  
 装置における画像編集方法であって、前記ホストコンピ  
 ュータから描画命令を入力する入力工程と、前記入力工  
 程で入力した前記描画命令を解釈して、確保した画像描  
 画メモリ上に画像データを描画する描画工程と、前記描  
 画工程で描画した前記画像データを前記ホストコンピュ  
 ータの画面メモリに表示する為に、前記画像データを前  
 記ホストコンピュータへ出力する出力工程とを有する。

【0016】

【作用】以上の構成によれば、サーバに描画命令を出力  
 し、出力した前記描画命令を解釈してサーバー上に確保  
 された画像描画メモリ上に描画された画像データを取得  
 し、取得した画像データを画面メモリに表示すること  
 で、クライアントでの画像描画処理をサーバーに代行さ  
 せることができる。

【0017】以上の構成によれば、ホストコンピュータ  
 から描画命令を入力し、入力した前記描画命令を解釈し  
 て、確保した画像描画メモリ上に画像データを描画し、  
 描画した画像データをホストコンピュータの画面メモリ  
 に表示する為に、画像データをホストコンピュータへ出  
 力することで、クライアントでの画像描画処理を代行し  
 画像編集処理できる。

【0018】

【実施例】図1は本発明の一実施例を示すマルチメディ  
 アサーバーのシステム構成を説明するブロック図であ  
 る。特に、センタサーバー11を介した既存サーバー1  
 0のアクセスを行う場合に対応する。

【0019】なお、本実施例に示すマルチメディアサー  
 バーにおいて、マルチメディア情報を扱うのに適したサー  
 ーバーは、「統合化サーバー」と「デジタル交換機」  
 とから構成され、ネットワーク上に仮想的な1個の統合  
 化サーバーを構築して各種のサービスを提供する。

【0020】本実施例において、「統合化サーバー」と  
 は、ネットワーク上に分散するサーバー群を論理的に統  
 合したもので、従来型の種々のサービスを提供する複数  
 の機能サーバーと、それらをコントロールする1個のセ  
 ンタサーバーとからなる。センタサーバーは専用サー  
 ーであってもよいし、クライアントが兼用するものであ  
 ってもよい。

【0021】クライアント（ホストコンピュータ）から  
 センタサーバーを見ると、ネットワーク上に分散する複  
 数の機能サーバー群が、統合化した1個の巨大なサー  
 ーのように見える。勿論個々のサーバーにも従来通り個  
 別にアクセスしてもよいが、センタサーバーに「プロセ  
 ススクリプト」を渡して以後の処理はセンタサーバーに  
 処理を任せた方が、クライアントは次の作用に素早く移  
 れるので効率的である。

【0022】図において、1はサーバーをアクセスする  
 「フロントエンド」としてのアプリケーションである。  
 ここで、「フロントエンド」を簡単に説明する。

【0023】もし、サーバーがクライアントと同じコン  
 ピュータに同居していたら、アプリケーションはサー  
 ーの機能を直接呼び出して利用できる。しかし、サー  
 ーがネットワークを介して、別のコンピュータ上にある  
 時は、簡単ではない。なぜなら、アプリケーションは、  
 通信プログラムを介してサーバーにアクセスしなければ  
 ならないからである。

【0024】これではアプリケーションのプログラマー  
 は簡単にはサーバーにアクセスするプログラムが書けな  
 い。そこで、サーバーのベンダーは、たとえそのサー  
 ーがネットワーク上にある場合でも、ローカルにある場  
 合と同じ呼び方でアクセスする仕組みを用意するのが普  
 通である。

【0025】その仕組みは、サーバーの一部の様なもの  
 で、通信機能を持ち、クライアントにインストールされ  
 る。この小さなプログラムの事を、サーバーに対するフ  
 ロントエンドと呼び、プログラマーは通信の事を何等意  
 識する事なしにネットワーク上のサーバーにアクセスす  
 るアプリケーションを作る事ができる。

【0026】ユーザがデータを入出力するのは、すべて  
 このアプリケーション1を通じて行う。2、4はクライ  
 アント／サーバー型の通信プログラム、3はローカルエ  
 リアネットワーク（LAN）である。5は前記アプリケ  
 ーション1に対して統合化された標準のアクセス環境を  
 提供するセンタサーバーである。アプリケーション1か  
 らはこのセンタサーバー5が全てのサービスを提供する  
 ように機能する。センタサーバー5は、既存の各種サー  
 ーを論理的に統合するのが基本機能なので、通常はサ  
 ー本来的なサービス機能は備えていない。そこで、サ  
 ーマネージャー6を通じて既存サーバー10にアクセ  
 スし、アプリケーション1からの要求に応える。7はク  
 ライアント通信プログラム、8はローカルエリアネット

ワーク (LAN) である。9はサーバー通信プログラムであるが、扱うデータとインタフェースの作りは通信プログラム2、4と同じではない。

【0027】既存サーバー10をアクセスする言語 (コマンド) は、それぞれのサーバーの構成との関係でサーバー毎の固有の言語を持っている。従って、アクセスするためのクライアント/サーバー通信プログラム7、9も独自のものである場合も多く、複数のサーバーを統合するためのセンタサーバー5と接続するクライアント/サーバー通信プログラム2、4とクライアント/サーバー通信プログラム7、9とは異なる場合もある。

【0028】また、LAN8に関しては同様の理由によりLAN3と同一である必要はない。例えばLAN3がAppleTalk (商品名) でLAN8がEthernet (商品名) であってもよい。既存サーバー10をコントロールするサーバーマネジャー6は、既存サーバー10のフロントエンドとして動作する訳であるが、クライアント通信プログラム7とのインタフェースは、やはり既存のAPI (Application Programming Interface) となる。

【0029】一方、センタサーバー5は全てのサーバーに共通のアクセス方法を提供し、サーバー/クライアント通信プログラム2、4を介して提供するAPIも共通のものとなる。

【0030】従って、センタサーバー5とサーバーマネジャー6との間でアーキテクチャの違いを吸収するためのプロトコル変換とデータ変換を行うことになる。

【0031】このため、センタサーバー5とサーバーマネジャー6は同一コンピュータの中にあって密接な関連の下で動作する。そして、センタサーバー5とサーバーマネジャー6との機能を統合した手段がセンタサーバーとしての機能である。従って、センタサーバー5は、機能を明確に区別する必要のある時はサーバー-サーバーと呼ぶ。

【0032】図2は、図1に示したセンタサーバー5を介した既存サーバーのアクセス方法を説明する図である。

【0033】図において、クライアント20~25は、共通のアクセス方法によりセンタサーバー26にアクセスすると、センタサーバーはクライアントからの要求に基づき、必要な機能サーバー27~30に個別のアクセス方法を通じてアクセスする。従って、各クライアント20~25は個別の機能サーバーA~Dへのアクセス方法はおろか、それらの存在すら関知する必要はない。

【0034】この様に構成されたマルチメディアサーバーにおいては、ネットワークを介して転送するセンタサーバーが各クライアントからの発行される所定のプロセススクリプトとデータとが一对となるメッセージを受信し、該受信したメッセージを解釈して、各機能サーバーに対する通信プロトコル変換及びデータ変換を施して各

機能サーバーに対する連続した処理を代行しながらそれぞれのプロセススクリプトとデータを前記ネットワークを介して転送するので、クライアントの各プログラムはセンタサーバーを複数の機能を実行させる場合であっても、センタサーバーにその複合化された機能処理に対応するプロセススクリプト及びデータを転送するだけで、所望の結果データを得ることが可能となる。

【0035】ここで「プロセススクリプト」を説明する。プロセススクリプトとは、通信プロトコルによってデータ通信をする基本単位で、コマンドがテキストで記述されている。コマンドとデータを一緒に送る事もできる。通常は機械と機械の間の通信を行う為に用いるので、コマンドはテキストであってもバイナリであっても差はない。しかし、テキストは人間が読めるので、機械が電子メールを介して人間に送り、その結果を電子メールで送ってもらうと言った変わった応用も可能である。

【0036】プロセススクリプトの最大の特徴は、通信プロトコル自身もスクリプトで送れるので、通信手順をダイナミックに変更しながら通信を成立させる事が可能となる。例えば、G4Faxの通信手順を思い出しみよう。サービスが、極めて厳格なプロトコルの上に構築されてるので、Faxを受け取ったら同じ文書を何箇所かに配信するなどという芸当は、標準プロトコルの中ではできない。

【0037】プロセススクリプトは、通信手順というより、サービスの手順を記述したものだから、通信方法や通信媒体はなんでも良く、基本サービスを理解できる端末なら、複合動作も簡単に指定できる。一例として、日本からアメリカのオフィスにあるプリンターにDTP文書を印刷する事を考えて見よう。普通はネットワークが繋がっていないので、この様な事はできない。新しい方式ならDTPアプリが吐き出す印刷コマンドをプロセススクリプトで包んで、電子メールで相手に届ける。相手がメールサーバーなら自動的に、そうでなければ手動で、受信したスクリプトをスクリプトマネージャに送ると、その内容が解析され、相手のオフィスのプリンターに印刷出力が出て来る。

【0038】また、各クライアントからの発行される所定のプロセススクリプトとデータとが一对となるメッセージを受信し、該受信したメッセージを解釈して、各機能サーバーに対する通信プロトコル変換を施して各機能サーバーに対するプロセススクリプトを前記ネットワークを介して転送するセンタサーバーを前記ネットワークに接続したので、クライアントの各プログラムはセンタサーバーを複数の機能を実行させる場合であっても、センタサーバーにその複合化された機能処理に対応するプロセススクリプトを転送するだけで、所望の結果データを得ることが可能となる。

【0039】さらに、センタサーバーからプロセススクリプトを受信した各機能サーバーは、プロセススクリプ

ト中のID情報に基づいてネットワークを介してクライアントからデータを受信するので、クライアント間と各機能サーバーにスクリプトとデータとを独立して転送することが可能となる。

【0040】また、ネットワークに接続されたセンタサーバーが各クライアントからの発行される所定のプロセススクリプトとデータとが一对となるメッセージを受信し、該受信したメッセージを解釈して、各機能サーバーに対する通信プロトコル変換及びデータ変換を施して各機能サーバーに対するプロセススクリプトとデータを前記ネットワークを介して転送し、交換機が各クライアントとセンタサーバーとの電話回線を交換するので各機能処理実行中に音声情報等のリアルタイム情報を並行してクライアント相互間でマルチセッションしながら複合情報を転送することが可能となる。

【0041】これにより、あらゆるサーバーのサービスを1種類の統一のとれた簡単なアクセスで処理手順を標準化し、シームレスで無駄の無いサーバー環境が提供でき、例えばFAXサーバーで受信した画像をOCRサーバーでコードに変換し、さらに、ファイルサーバーでデータベースに蓄積するといった一連の複合作業を、1つのスクリプトで連続して処理することが可能となる。従って、従来同様のクライアント側で処理に必要とされるような、クライアント側でFAXフロントエンドとOCRフロントエンドとデータベースフロントエンドの計3本のフロントエンドプログラムを起動し、順次作業を進める、カット&ペーストや一時ファイルへのアクセスの繰返し処理を大幅に軽減することができる。

【0042】また、ネットワーク上に分散している独立サーバーを仮想的に統合できるため、サーバー毎の互換性をクライアントが意識することなく、1本のアプリケーションから簡単に、かつ多くのサーバー機能もしくは複合的なサーバー機能を利用することが可能となる。

【0043】さらに、従来、個別の企業／業界において、ばらばらに開発されてきたサーバー装置へのアクセス方法を統合化し、ネットワーク上の既存のばらばらなインタフェースを有するサーバーに1つの統合されたアクセス方法でアクセスする手段を提供することが可能となる。以下、それぞれの実施例に分けて詳述する。

【0044】図3は本発明に係るマルチメディアサーバーにおけるメッセージの構造を説明する図である。

【0045】図3に示すように、本実施例におけるメッセージは、プロセススクリプトフォーク（プロセススクリプト）とデータフォーク（データ）とから構成されている。また、プロセススクリプトとデータは、それぞれ共通の内容を持つタグエレメント（タグ）を備えている。

【0046】このタグには、メッセージが作成された「時間」、メッセージが消去されるべき「ライフタイム」、一連の「ID番号」、データの「種類」、データ

を作成したアプリケーションの「サイン」等が記載される。

【0047】この内「ID番号」は必須であるが、他はオプションである。

【0048】図4は本発明に係るマルチメディアサーバーにおける第1のプロセススクリプトの送出手順を説明するブロック図である。

【0049】通常、クライアント41、センタサーバー43、機能サーバー44は、全て同一のLAN上に存在する場合が多い。そして、機能サーバー44が本発明に準拠したプロセススクリプトを理解できるサーバーの場合には、センタサーバー43がクライアント41にその事を指示し、クライアント41はメッセージ42をプロセススクリプト45とデータフォーク（データ）46とに切り放して、それぞれセンタサーバー43と機能サーバー44に別々に送信することができる。センタサーバー43は、受信したプロセススクリプト45に必要な編集を施した後、機能サーバー44に新たなプロセススクリプト47を送り届ける。機能サーバー44にとっては、クライアント41もセンタサーバー43も共にクライアントであるから、別々に届いたプロセススクリプト47とデータ46を結合し、データに必要な処理を施して結果のメッセージ48をLANを介してクライアント41に返す。

【0050】なお、本実施例において、プロセススクリプトとは、センタサーバー43が機能サーバー44群を使って一連の仕事をするための手順を記述したプログラムリストであり、センタサーバー43はそれに基づいて一連の作業を実行し、最終結果のメッセージ48だけをクライアント41に返すので、クライアント41の負荷は、従来型の機能サーバー群に個別にアクセスを繰り返していくよりも極めて小さくなる。

【0051】プロセススクリプトは、クライアント41のデバイスドライバーが自動生成することもあり、該プロセススクリプトはクライアント41にとっては、通信プロトコルの一連の関数コールの集りにしか過ぎず、センタサーバー43にとっては一連の通信プロトコルの中から生成されるスクリプト言語であり、機能サーバー44群にとっては通信プロトコルそのものである。

【0052】プロセススクリプトの特徴として、従来のコンピュータ通信の形態を大きく分けると、

リアルタイム処理  
バッチ処理  
の2つがあった。

【0053】バッチ処理は、大型コンピュータ（メインフレーム）で処理する時に良く使われた手法で、大勢のユーザ（クライアント）が処理を集中させた場合、ジョブをキューに順番に溜め込み、順番に処理してユーザに返すというやり方である。結果がいつ出てくるのか分からないという欠点があり、どうしても進捗を見たい場合

10

20

30

40

50



には、キューを覗いたりしなければならなかった。

【0054】リアルタイム処理は、プロトコルのやり取りによって何時でも進捗が分かる仕組みが用意されており、そのプロセスの中で必要な処理が実行される。

【0055】一見便利に見えるリアルタイム処理であるが、これにはネットワークもリアルタイム系でなければならないという条件が付く。例えば、ネットワークがインターネットの様な「ストアード・フォワード」型の場合、ハブを何箇所か中継して行くと、その度に数分の遅延が生ずる。この様なネットワークでは、リアルタイム処理が使えない。

【0056】プロセススクリプトは、この様な「ストアード・フォワード」型のネットワークに最適な通信方法で、リアルタイム処理とバッチ処理の中間的な特性を持っている。スクリプトとは「記述する」という意味で、コマンドがテキストで記述されており、その分プロトコルが簡単になる。

【0057】リアルタイム処理の例を、図5に示す。この図では、ユーザが「データを送るよ」というコマンドを発行し、サーバーから「いいよ」という返事をもらった後、「データ」を送り、「完了」という返事に対し、「了解」という送達確認を送っている。この様なプロトコルが使えるのは、間に何もハブを介さないユーザとサーバーのエンドツーエンドの通信だからである。

【0058】これがもし図6の様に、ハブ1、2、3を介在する通信の場合どうなるかと言うと、同図の様に「データを送るよ」から「いいよ」まで順番に遅延が蓄積して行き、「データ」送信にたどり着くだけでも大変で、そのうちどこかの通信接続が（タイムアウト等で）切れてしまう。全く実用にならない。

【0059】図7はストアードフォワード型のネットワークを考慮したバッチ処理の例である。「データを送るよ」に対して「いいよ」の返事は、近接したハブからもらっている。この様にすれば、比較的早く「データ」を送れる。しかし、この方法でも、途中で何らかの通信エラーが発生した時の回復ルーチンは、気が遠くなるほど複雑である。「データを送るよ」が途中で紛失した場合ならなんとかなるだろうが、「いいよ」が紛失したら、その回復は多分不可能であろう。

【0060】図8はプロセススクリプトを用いた通信方法の例である。「処理して返事しろ」というコマンドと「データ」がセットになって順に送られる。処理結果も「処理結果だよ」というコマンドと処理済みの「データ」がセットで送られるので、このネットワークではこれより早い通信方法は無い。通信異常でプロセススクリプトが紛失した場合でも、コマンドとデータがセットになっているので回復も容易である。

【0061】図9は本発明に係るマルチメディアサーバーにおける第2のプロセススクリプトの送出手順を説明するブロック図であり、特に、機能サーバーが既存のサ

ーバーである場合のアクセス方法を示したものであり、図4と同一のものには同一の符号を付してある。

【0062】この図において、メッセージ42のプロセススクリプトとデータは分離せずにセンタサーバー43にリクエストとして送られる。リクエストされた機能がセンタサーバー43に無い場合には、その機能を代行してくれる機能サーバー44を捜し、それに既存のアクセス方法でクライアント41のリクエストを伝え結果を受け取る。その結果はメッセージ48としてクライアント41に返される。〈通信プロトコル〉マルチメディアサーバーにおけるセンタサーバーの役割の一つは、種々のサーバーのバラバラなアクセス方法を標準化することにある。アプリケーションは、標準化されたインタフェースを持つセンタサーバーにアクセスすれば、そこから個々のサーバーへのアクセスはセンタサーバーが代行してくれる。標準化する項目は、大きく分けると、データの標準化とアクセス方法の標準化の2つある。

【0063】データの標準化は一本化ではなく、複数化である。特定のアプリケーションに依存しない、業界標準のフォーマットで標準化し、かつ、標準化されないデータのやり取りも許容する。

【0064】主なフォーマットとして現在使用されているものは、QuickDraw、GDI、RTF、TIFF、PICT、Bitmap、PostScript、EPS、G3/G4、PCL、HP-GL、ANSISQL、ASCII Text、UNICODE Text、Binary Data等（登録商標及び商品名を含む）である。

【0065】一方、アクセス方法の標準化は一本化に近い。統一されたアクセス方法により、さまざまなサービスを統一的に簡単に利用できる。

【0066】ただし、業界標準のアクセス方法はサポートしない訳にはいかない。一方、図3で説明したように、伝送する内容はメッセージ、すなわち、「プロセススクリプト」と「データ」とがセットになっている。

「データ」は、処理されるべき情報そのものであり、「プロセススクリプト」はその情報をどう扱うかを記述したコマンドシーケンスである。「プロセススクリプト」と「データ」のセットがサーバー間を伝送され、必要な処理がなされていく。「プロセススクリプト」の受渡し方法を標準化することにより、さまざまなサーバーに同じアクセス方法でアクセスできる。

【0067】また、プロトコルの異なるサーバーに対しては、センタサーバー43がゲートウェイとして機能する。例えばクライアント41からFaxサーバーにアクセスするための関数コールが図10の(a)に示す内容が発せられ、その結果、図6の(b)に示すようなスクリプトがセンタサーバー43に渡される。

【0068】図10は図9で用いるプロセススクリプトの例である。(a)はプロセススクリプトを生成する時

10

20

30

40

50



の関数。(b)は実際にネットワーク上に送出されるスクリプト部分である。この場合、G3ファクシミリを例に取っているのが、データとしてはMMRデータが流れるが、省略してある。

【0069】MM\_openでサーバーのアドレス(server)とサービス種別(fax\_service)を指定する。

【0070】MM\_sendでfax\_serviceの形式はG3である事を指定する。送信先Fax番号(destination)もセットする。

【0071】MM\_dataで送信データバッファのサイズ(length)と中身(content\_buffer)を送る。

【0072】MM\_closeでこのサービスを終了させる。プロセススクリプトは柔軟なので、この時はリアルタイム系に近い使い方をしている。

【0073】図11では、図10で簡略化して書いたシーケンスを詳細に説明している。

【0074】アプリケーション、クライアントとセンタサーバーとの通信プロトコルは、図11に示す手順に従うものとする。

【0075】なお、アクセスシーケンスとして、以下の2種類の方式がある。

【0076】第1は、ストアードフォワード方式で、蓄積交換であり、プロセススクリプトとデータのペアを伝送し終わったら、処理結果の成功/失敗に関係なく通信を一旦終了し、処理の最終的な結果は、改めて通信を再開して調べる。サーバーに作業を委託し終わったら、クライアント側では次の作業に移れるので、資源の利用効率が上がる。

【0077】第2は、リアルタイム方式であり、最終的な処理結果が確定するまでは通信の接続を確保し続ける方式で、クライアントの作業が拘束される欠点はあるものの、処理の信頼性は高い。

【0078】以下、本実施例におけるデジタル交換機の機能処理について説明する。

【0079】本実施例のデジタル交換機は、公衆回線交換網および構内回線交換網を接続制御する交換機で、センタサーバー43からの接続指示に基づいて呼制御を行える。LANを経由した遅延のある接続がなされている複数のクライアントの間に、構内回線交換網を経由した遅延のない第2の接続を行うことにより、リアルタイム性を要求される情報(音声)等を伝達できる。

【0080】図2は本発明におけるセンタサーバーと交換機を結合したマルチメディアサーバーによるシステム構成を説明するブロック図であり、図2の応用例である。

【0081】図12において、61~65は各種の機能サーバー、66~70はクライアント、71は各機能サーバー61~65を統合するセンタサーバー、73はロ

ーカルエリアネットワーク(LAN)、72は内線/外線の電話回線を交換する交換機(PBX)である。LAN73を介したデータ転送は、パケット化されて伝送されるため、音声や動画等のリアルタイム系の伝送には適さないが、全てのクライアントが常に接続されているという特徴を生かして、接続制御の不要なデータ転送や、同報通信に適する。

【0082】一方、PBX72は音声や動画等のリアルタイム系の伝送に適する。LANとPBXの特徴を生かして、パソコン会議システム(PC会議)を構築することができる。いま、クライアント66から67にアクセスしてPC会議を行う場合を考える。クライアント66はノード75を介してセンタサーバー71に対し、クライアント67とPC会議を開始するための開始命令スクリプトを発行する。センタサーバー71は、クライアント67のマルチメディアクライアントデーモン(MCD)と接続しPC会議の開始を指示するスクリプトを発行する。ここで、デーモンとはバックグラウンドで常に走っているプログラムのことをいう。そのスクリプトにはPC会議のID番号が書かれているので、以後そのID番号宛にクライアントからスクリプトを送り合うことにより、各クライアント間でデータ通信が可能となる。

【0083】一方、同時に、センタサーバー71は、制御線74を介してPBX72にアクセスし、クライアント66と67の近く(または内蔵)の電話機同士を接続し、音声の伝送を可能にする。この様にして各クライアント66、67のオペレータはコンピュータの画面上の共通の画面を見ながら音声で会話をし、PC会議を行う事ができる。PC会議は、1対1の接続である必要は無く、複数のクライアントを同時に接続する事もできる。

【0084】これにより、パケット伝送を主体とし、リアルタイム系のデータ伝送を苦手とするLANと、音声/動画の交換を得意とするが、同報通信やインテリジェントな制御を苦手とする交換機を有機的に結合し、市販のパソコンに何等特別なオプションパーツを付加接続することなく、パソコン会議システムを構築することも可能となる。

【0085】以下、高速データハイウェーを利用したサービス機能について説明する。

【0086】なお、本実施例では、今日比較的入手し易いコンポーネントを想定してシステム構築を考えたと、FDDI(Fiber Distributed Data Interface:米国ANSI規格)の様な100Mbpsないしそれ以上の伝送レートを有するネットワークが利用できる場合には、データパケットと音声回線をマルチプレックスしても良い。

【0087】同様に本実施例では、交換機PBXに接続する公衆網として、ISDNを想定しているが、B-ISDNであっても構わない。また、米国のSMDSCラ

接続しても構わない。

【0088】また、多少速度を犠牲にしても構わない場合には、UNIXのSLIP (Serial Line IP) の様に、LAN間接続をISDN網を介して接続しても構わない。また、現状の技術でも、ISDNルータを用いればISDN網を介してLAN間接続が可能である。

【0089】以下に、本発明が適応可能なサービスの例を示す。

【0090】(分野) - (サービス)

印刷-カラー印刷

スキャナー-カラースキャナ

OCR-OCR、ファイルキーパー、伝票処理

翻訳-日英

ファイルシステム-NFS

データベース-テキスト、静止画、動画、音声

会議システム-共用ウインドウ+リアルタイム音声

メール-テキスト、静止画、動画、音声

投稿システム-テキスト、静止画、動画、音声

画像処理-CMM、フィルタリング、画像処理アクセラレータ

Fax-Fax送信/受信/配信

MHS-テキスト、Fax、CATS

予約システム-会議室予約等である。

【0091】以下、OCRサービスを例としてサービス機能処理について従来と本実施例とを対比して説明する。

【0092】いま、10枚の原稿をスキャンし、それをOCRにかけて文字コードに変換し、それをテキストファイルとしてデータサーバーのディスクにセーブする作業を考える。

【0093】従来では、(スキャナで原稿のスキャンしてクライアントに伝送) (イメージデータをOCRサーバーに再転送OCRに掛けてテキストデータに変換) (テキストデータをクライアントに伝送) (テキストデータをデータサーバーのディスクに転送) を10回繰り返す。

【0094】一方、本実施例では、クライアント→センタサーバーにプロセススクリプトを送信(スキャナで原稿のスキャン→OCRサーバーに伝送) (OCRに掛けてテキストデータに変換) (テキストデータ→データサーバーのディスクに転送) センタサーバー→クライアントに結果を知らせる。

【0095】従来例では、クライアントが介在するデータ転送が4回必要で、通常のパソコンではハードディスク容量の制限から、10枚もの原稿を一気にスキャンする事はしないので、最悪40回のデータ転送が必要になる。それに対してクライアントが介在するデータ転送は2回で済み、その他はすべてサーバー同士の通信である。全体のデータ転送も減っている。クライアントが関

係しないデータ通信/処理はカッコで示してある。データ転送の回数が減るという事は、コンピュータを操作する回数も減る事になるので、作業の大幅な自動化が実現される。

【0096】以下、スキャンサービスを例としてサービス機能処理について従来と本実施例とを対比して説明する。

【0097】A3サイズ of 原稿をネットワークスキャナで400dpiフルカラーでスキャンし、それに色処理して、プリンタサーバーに印刷する作業を考える。

【0098】従来例では、原稿のスキャン→クライアントに伝送イメージデータ→色処理後のイメージデータ→プリンタサーバーに伝送印刷

一方、本実施例では、クライアント→センタサーバーにプロセススクリプトを送信(原稿のスキャン) (イメージデータ→色処理) (色処理後のイメージデータ→プリンタサーバーに伝送) (印刷) センタサーバー→クライアントに結果を知らせる。

【0099】通常のパソコンではメモリ容量の制限があるので、実は、従来例というのは机上の計算値でしかなく、実際には96MByteもの巨大なメインメモリを管理できるOSは事実上存在しないに等しい。

【0100】以下、メールサービスを例としてサービス機能処理について従来と本実施例とを対比して説明する。

【0101】動画ファイルをサーバーに送って動画のプレゼンテーションを実行する場合。

【0102】従来例では、

巨大な動画ファイル→サーバーに伝送

一方、本実施例では、クライアント→センタサーバーに動画スクリプトを送信(センタサーバー機能サーバー間通信) センタサーバー→クライアントに結果を知らせる。

【0103】動画ファイルは一般的に極めてサイズが大きくなる。数10分程度の映画で1ギガバイトを越える事もある。

【0104】このようなサイズのデータを扱うためには、従来のクライアントのメモリやハードディスクを増設し、場合によってはCPU本体も高速な物に交換する必要がある。

【0105】しかし、本実施例では、サーバーが動画データを記憶してくれるのでクライアントのメモリやハードディスクを増設する必要は無く、CPUも遅い物で構わない。ネットワークを介してマルチメディアサーバー(センタサーバー)にプロセススクリプトを発行すれば、動画の再生はサーバーが代行してくれる。

【0106】以下、メールサービスを例として会議システム機能処理について従来と本実施例とを対比して説明する。

【0107】複数のクライアントを同時に接続し、同じ

ファイルをオープンしてどこからでも書き込める電子黒板を用意し、同時に交換機にアクセスしてそれぞれの内線電話同士を接続し、パソコン会議を行う。

【0108】従来例では、音声の入らない、テキストベースの会議システム程度一方、本実施例では、クライアント→センタサーバーに会議用プロセススクリプトを伝送（センタサーバー機能サーバー間通信）（クライアントとクライアント間のパソコン会議）（クライアント→センタサーバーに終了プロセススクリプトを伝送）伝送センタサーバー→クライアントに結果を知らせる。

【0109】以下、図13を参照しながら本発明に係るマルチメディアサーバーを画像描画アクセラレータとして機能させる場合について説明する。

【0110】図13は本発明に係るマルチメディアサーバーを画像描画アクセラレータとして機能させる場合の概念を説明するブロック図である。特に、本実施例では、マルチメディアサーバーをアップルコンピュータ株式会社のコンピュータ（登録商標：Macintosh）の画像描画アクセラレータとして用いた場合の例である。

【0111】例えば、B5で400dpiフルカラー程度以上のビットマップ画像の編集を行う為には、メインメモリが数十メガバイトも必要になり、従来のパソコンでは不可能であった。最新のOSの提供する仮想メモリ機能を利用しても、処理速度の点からA4で400dpiフルカラー程度が限度であろう。

【0112】一方、本実施例のマルチメディアサーバーをネットワーク上の仮想メモリとして利用する事により、A3サイズ程度以上のフルカラー高解像度画像の編集を可能にする。ここで、仮想メモリを説明する。

【0113】極めて巨大なメモリを利用する方法。実メモリとは、通常は半導体メモリ（あるいは主記憶）の事を言い、要求された巨大なメモリサイズのごく一部しか存在しない。それに対して、仮想メモリとは、ハードディスクやネットワーク上のサーバーのメモリ（あるいは補助記憶）の事を言い、アクセス速度は遅いが極めて巨大な記憶空間を有する。

【0114】主記憶と補助記憶をうまく組み合わせ、アプリから見たら、全メモリ空間が一樣でシームレスな空間であるかのように見せる方法を仮想メモリと言う。

【0115】アプリが主記憶に無い部分をアクセスしに行くと、メモリコントローラはエラーを検知するので、それまであった主記憶の情報を補助記憶に退避させ、逆に補助記憶から必要な情報を主記憶に呼び出し、かつ、その主記憶のアドレスを要求されたメモリ空間にスワップさせる事により、アプリから見たらあたかも巨大なメモリが実際に存在するかに見える。

【0116】具体的には、アプリケーション101は、モニタに何かを表示させる時には、必ずOSの一部である描画管理プログラム（商品名QuickDraw）1

02に描画命令を送り、画面メモリ109へのラスターライズを依頼する。アプリケーションが画像メモリを直接アクセスする事は無い。

【0117】一方、ビットマップ画像をモニタに表示させる場合には、通常アプリケーションの責任で描画用バッファメモリを確保し、そこにバックグラウンドで描画した後、QuickDraw102にメモリ転送の依頼を出す。

【0118】従って、描画用バッファメモリはアプリケーションの管理下であり、それを仮想メモリにして実メモリをハードディスク上に設定する事は従来から行われていた。

【0119】本実施例においては、実メモリをネットワーク上のマルチメディアサーバー上に設け、画面がアップデートされた時にローカルメモリに部分コピーを行う。

【0120】103はマルチメディアサーバーの機能をクライアント側でアプリケーションに引き渡す為のAPI（Application Programming Interface）である描画コマンドハンドラである。アプリケーション101がこのAPIに対してQuickDrawに対するコールと同じパラメータを渡してネットワーク上のサーバーに確保したメモリに描画する。

【0121】アプリケーションは必ずしもマルチメディアサーバーを意識して設計されている訳ではないので、アプリケーション101がQuickDraw102への関数コールをフックして描画コマンドハンドラ103に強制的に制御を渡しても構わない。QuickDrawの関数コールは全て例外処理で行っており、その処理を分岐させる事はたやすい。この場合、画像メモリに対するドロー描画コマンドはQuickDrawにスルーさせる。マルチメディアサーバーを意識して設計されたアプリケーションの方が描画効率が高い事は言うまでもない。

【0122】一方、メモリに対するビットマップ描画コマンドはクライアント104、サーバー105を介してPDLラスターライザー106に送られる。ビットマップ描画コマンドはここでビットマップデータにラスターライズされ、マルチメディアサーバー内のリモート実メモリ108に描画される。リモート実メモリ108は1ページ分のビットマップデータを全て記憶できる容量がある。それに対してローカル仮想メモリ107の容量は限定されたものである。

【0123】ローカル仮想メモリ107は、仮想的にリモート実メモリ108と重なり合うものであるが、実際にメモリが割り当てられている部分は、現在編集中の限られた領域のみである。編集中の領域が変更になれば、新しい領域の画像データをリモート実メモリ108からコピーして来る。

10

20

30

40

50

【0124】これにより、通常ではメモリ容量等の制限から、A3、400dpiフルカラー画像の編集等、従来ならば不可能な画像編集処理を安価な端末からネットワーク上の資源を利用して処理することができる。

【0125】図14に本発明による画像描画アクセラレータのブロック図を示す。点線から右がサーバー側、点線から左がクライアント側である。図13と同じ機能要素には同じ番号を付与してある。

【0126】例えば、クライアントがMacintosh（商品名）の場合を例に取って説明すると、普通の使用方ではアプリケーション101が画面に描画する場合、*grafport*と呼ばれる画面メモリ管理システム109をコールし、描画ポートを確保する。*grafport*が確保する画面バッファメモリはビットマップ系であり、その解像度は72dpi（ドット/インチ）である。

【0127】アプリケーション101はQuickDraw（商品名）と呼ばれる一連の画面描画関数をコールする事ができ、通常それはToolBoxコール（OSのシステムコール）の形でOS102U制御が渡される。OSは受け取った画面描画関数コールをビットマップにラスターライズして*grafport*に描画する。

【0128】本発明では、QuickDrawのToolBoxコールをフック（盗んで）して一旦ネットワーク描画アクセラレータ110（*net\_grafport*）に制御を渡し、それが改めてToolBoxコールを発生させている。ネットワーク描画アクセラレータ110は描画コマンドハンドラ103とクライアン通信プログラム104を包含した部分を説明している。

【0129】ここで、クライアン通信プログラム104は、単に通信手段を提供する通信プログラムであるが、クライアント側と言うと、図の点線から左すべてを差して呼ぶ。サーバー側とサーバーの呼び方についても同様で、通信プログラム部分だけを指す場合とサーバー側と言うと点線から右全体を指す場合とがある。

【0130】さて、ネットワーク描画アクセラレータ110は、アプリケーション101からのQuickDrawのToolBoxコールをプロセススクリプトに変換してLANを介してネットワーク描画アクセラレータサーバー（サーバー側）にも伝える。それを受信した画像処理サーバ111は、例えば、400dpi（ドット/インチ）の高精細の解像度で高精細描画システム108（*server\_grafport*）にクライアントと同じ画像を描画する。ローカル画面メモリ（画面メモリ管理システム109が管理するバッファメモリ）への描画が完了すると同時に、サーバー側では高精細な画像の描画が完了しており、必要ならそのまま印刷出力もできるし、フィルムに焼く事もできる。

【0131】図15にネットワーク描画アクセラレータ110のブロック図を示す。クライアント側に関しては

全てソフトウェアで実現する事も可能であるが、処理を高速化させる為にアクセラレータボードによるハードウェアで実現する事もできるこのアクセラレータはボードの形でクライアントのコンピュータのバス（vバス）209に挿入する。バスI/O206はボード内部の内部バス205と外部バス209との接続とアービトレーションを行う為のバスI/Oである。内部バス上にはCPU201、ROM202、RAM203、LANI/F204があり、マイクロコンピュータを形成している。

【0132】割込ハンドラ207は、クライアントコンピュータ本体の割り込み信号線210をボード上のCPU201に導入する為のものである。BUSINT信号211は、I/O208を介してマザーボードに出力され、クライアントコンピュータ本体のCPUにバスの使用を要求するための制御線である。BUSACK信号212は、バスが解放されたかどうかを知らせる制御線である。

【0133】アプリケーションが描画コマンドを発行すると、何らかのQuickDrawルーチンがコールされ、割り込み（INT）が発生するから、その信号は割り込み信号線210を介して検知する事ができる。その割り込み信号に基づき、CPU201はI/O208をイネーブルにし、BUSINT信号211をグラントしてマザーボードのCPUにバス解放要求を出す。バスが実際に解放されてトリステート状態になったら、BUSACK信号が帰されてCPU201はバス209が使える状態になった事を知る。そこでBUSI/O206をアクセスして内部バス205と外部バス209を結合し、所定の外部メモリ（図示せず）に書かれている割り込みジャンプテーブルのポインタをフェッチして来る。ポインタの指し示すメモリの内容を読むと、割り込みの種類が判別できるから、QuickDrawルーチンに関する割り込みであった場合には、RAM203上でQuickDrawコマンドをプロセススクリプトに変換してそれをLANI/F204からサーバーに伝送する。

【0134】サーバー側で、全画面のリモート実メモリに描画された内容のミラーイメージが、ローカル仮想メモリ107に反映される。実際には、画面メモリ109と同じ大きさのローカル実メモリ112の部分のみがコピーされる。アドレスコンバーター113は、外部バス209から見た実メモリ112が、あたかも仮想メモリ107の一部であるかの様に見せかける為のアドレス変換を行う回路である。仮想メモリ107の原点（左上）の座標が、X、Yであり、実メモリ112の矩形領域の座標が、x、yであるなら、アドレスコンバーター113は外部バス209からアクセスされるアドレスに対し、X-x、Y-y、の演算を実行する。

【0135】マザーボードのCPU（図示せず）がCRTに表示している部分、すなわち実メモリ112の内部

だけで演算している場合は、仮想メモリ上の実メモリの位置関係は変化しない。しかし、スクロール等で実メモリの外側をアクセスしようすると、割り込み信号114を発生させ、CPU201に割り込み信号を与える。CPU201は割り込みの原因を調べ、LAN204を介してサーバーに対して新しい座標位置のビットマップ画像を送信する事を要求すると共に、アドレスコンバーターをセットしなおし、実メモリの相対位置を変更する。割り込み信号114は、実際にはアドレスコンバーター113が生成する。

【0136】以下、図16を参照しながら本発明に係るマルチメディアサーバーにおけるローカル仮想メモリ処理動作の概要について説明する。

【0137】図16は発明に係るマルチメディアサーバーにおけるローカル仮想メモリ処理の概要を示すチャートである。例えばMac OSによるQuickDrawのアクセラレータ処理について説明する。

【0138】なお、本実施例のローカルメモリは、本来アプリケーションがOSに要求して実メモリとして確保する代りに、ネットワークで接続されたサーバー上に確保した実メモリを仮想的にリモートのメモリであるかのように処理する。従って、ローカル仮想メモリは、ローカルなクライアントマシン上に仮想的に存在し、実メモリはリモートのサーバー上に実在する。

【0139】具体的には、アプリケーションが描画を開始する時、Mac OSに対してメモリ確保コマンドとして、例えばNewPtr (New Pointerの略でメモリ領域を確保するコマンド) を発行する。描画コマンドハンドラは、そのコマンドをストール (盗んで) してプロセススクリプト (Pスクリプト) に変換し、クライアント/サーバーの通信路を介してPDLスタライザ/サーバー (以後、リモート描画エンジン) にプロセススクリプト (Pスクリプト) を送る。

【0140】リモート描画エンジンは、リモートのOS (図示しない) にmallocコマンド (memory allocateの略でNewPtrと同じ働きをする) を発行して実メモリを確保するとともに、プロセススクリプトをクライアントに返す。そのプロセススクリプトは、描画コマンドハンドラからアプリケーションに対し、メモリのポインタとして返されるので、アプリケーションからはあたかもローカルに実メモリがあるかのように見える。

【0141】次に、アプリケーションは、ローカル仮想メモリに何かを描画しようとして、Mac OSに「描画コマンド」を発行する。その「描画コマンド」は同様に描画コマンドハンドラにストールされ、プロセススクリプトに変換され、「リモート描画エンジン」に送られてリモート実メモリ上に実際に描画される。これは、400dpiの高解像度で行われる。描画コマンドハンドラは、クライアントのモニタ上にも描画する必要があるの

で、クライアントのMac OSに対しても描画コマンドをエミュレートして伝送する。この場合は、72dpiの低解像度で描画される。

【0142】一方、描画の結果はプロセススクリプトに変換され、同様にクライアントに戻される。このプロセスは、必要な回数だけ繰り返し実行される (図中の太線の流れに対応する)。

【0143】描画が終了して印刷をする場合、本実施例によらなければ、ローカルのクライアント上で400dpiの高解像度でラスタライズし、その膨大なデータをプリンタに伝送することになるが、本実施例によれば、描画が終了した時点で高解像度のラスタライズは「リモートエンジン」上で完了しているので、膨大なデータを送ることなしに、直ちに印刷を開始できる。従って、伝送されるのは、「印刷コマンド」のみである。

【0144】なお、完成したビットマップの高解像度の画像データを2次記憶装置等 (図示しない) にセーブする場合にも、「セーブコマンド」を発行するものの、それによって「リモート描画エンジン」から送られてきた「Data」を直接2次記憶装置等にセーブすれば、ローカルに実メモリを備える必要はなくなる。

【0145】以下、図13のシステム原理図と図17のフローチャートの両方を参照しながら、画像描画アクセラレータの動作を詳細に説明する。

【0146】図17は画像描画アクセラレータ110の動作フローチャートであり、中央から左半分がクライアント側、右側がサーバー側の動作を示す。図では、クライアント/サーバー間の一般的な接続動作 (例えばAppleTalkやTCP/IPなどの接続) は完了しているものとして説明している。なお、図16の左半分のフローチャートに示すプログラムは、クライアント側のネットワーク描画アクセラレータ110のROM202或いはRAM203に格納されており、CPU201により実行される。また、図16の右半分のフローチャートに示すプログラムは、サーバー側の不図示のROM或いはRAMに格納されており、不図示のCPUにより実行される。

【0147】クライアントのアプリケーションが描画作業を実行する前には、その結果を保存するメモリを確保しに行く。その動作はOSに対する要求動作であり、一般的にはシステムコールと呼ばれる。Macintoshの場合にはツールボックスコールという。描画コマンドハンドラ103を介してCPU201はその関数コールをフックして、サーバー側に実メモリを確保する様に要求する。それがステップS12と23のやり取りに示される。

【0148】同時に描画コマンドハンドラ103を介してCPU201は、ステップS13に示す様に、ローカルマシン上にも、画面メモリ用の最低限のバッファメモリを確保する。このローカルマシンの画面は72dpi

(ドット／インチ)なので、バッファメモリのサイズは極めて小さい。

【0149】次に、リモート実メモリ108と同じサイズのローカル仮想メモリ107のポインターのみを確保し、ステップS23で帰された実メモリへのポインターで、上書きする。さらに、ステップS13で確保した画面メモリのポインターを、仮想メモリ107空間の適当な部分に割り当てる。これによって、巨大なリモート実メモリ108に描画された絵の一部分がクライアントの画面に表示される様になる。

【0150】その後、描画コマンドハンドラ103を介してCPU201はイベント待ち(ステップS15)になり、もし何らかの描画イベントが発生すると(ステップS16)その描画コマンドをサーバー側に送信すると共にローカルマシンの画面メモリにも同じ描画を実行して(ステップS17)画面表示する。描画コマンドを受信したサーバー側はPDLラスターライザ106を介して不図示のCPUが実メモリ108に高精細で描画する(ステップS24)。

【0151】先にも述べた様に、ローカルマシンの画面には巨大なリモート実メモリ108に描画された絵の一部分が表示されているので、イベントとしては描画コマンドだけではなく、画面を上下左右にスクロールさせるスクロールイベント待ちになる(ステップS18)。この場合、ローカル仮想メモリ107の表現するメモリ空間の一部に存在する様に割り当てられているローカル画面メモリ109の矩形領域を表すパラメータが、スクロールさせる分だけ変更になる(ステップS19)。

【0152】ローカル画面メモリ109をローカル仮想メモリ107空間内で移動させると、ローカル画面メモリに書かれている絵と、リモート実メモリ108に描画された同じ位置の絵とが食い違って来る。そこで描画コマンドハンドラ103を介してCPU201はサーバーに対して新しい矩形領域のビットマップ画像を送る様に要求する(ステップS25)。

【0153】サーバーの不図示のCPUは指定された領域のビットマップ画像情報をクライアント側に送り(ステップS26)、クライアント側のCPU201はそれをローカル仮想メモリ107にコピーする。すなわち、結果としてその画像情報はローカル画面メモリ109上に書きされ、表示画面は新しくなる。

【0154】その後はイベント待ちを繰り返す。

【0155】アプリケーション101がビットマップ画像を編集する場合、OSの描画機能102を用いてローカル実メモリ(図示せず)上で画像編集し、その結果を画面メモリ109を介してモニター(図示せず)に表示する。しかし、本発明で用いるローカルメモリは、その一部しか実体として持たない仮想メモリ107である。これによって巨大なメモリを用意する必要がなくなった。この場合、図13の×印の様に本来全ての描画コマ

ンドがOSのラスターライザ102に渡されていた所をストールして描画コマンドハンドラ103に導き、必要に応じてラスターライザ102でローカル仮想メモリ107と画面メモリ109を間接的に制御する。

【0156】ストールされた描画コマンドは、クライアント通信プログラム104、サーバー通信プログラム105を介してサーバー側のラスターライザ106に伝達される。そして高精細なリモート実メモリ108にビットマップ画像が描画される。リモート実メモリ108上にラスターライザされた画像は、そのまま印刷できる。

【0157】ローカル仮想メモリ107は、画面メモリ109と同じサイズのメモリで、リモート実メモリ108の一部分に相当する。ネットワーク画像処理を行う場合、まず104と105の間で一般的なクライアント／サーバーコネクションを張ったあと、アプリケーションが要求する画像メモリを確保する。メモリ確保要求は描画コマンドハンドラ103がフェッチし、実際にはローカルマシン上には仮想メモリ107が確保され、本当のメモリの実体は、ステップS12と23のやり取りにより、サーバー上のリモート実メモリ108として確保される。仮想メモリ107及びリモート実メモリ108の仮想的な大きさは一致しており、扱う原稿サイズ毎に異なる。

【0158】画面メモリ用のバッファも同時にステップS13でローカルマシン上に確保される。その後はアプリケーション101は描画入力イベント待ちの状態になり、すなわち描画コマンドハンドラ103も入力待ちになる。これはステップS15に示される。もし操作者がマウス等を操作して、何らかの描画のコマンドを入力すると、それがもし描画イベントならば描画コマンドがローカルマシンのラスターライザ102とサーバーのラスターライザ106の両方に送られて、画面メモリ109とローカル仮想メモリ107及びリモート実メモリ108の3箇所にそれぞれCPU201、サーバーの不図示のCPUによって描画される。このステップを16、17、24に示す。

【0159】もし、先の描画イベントが画面のスクロールの場合には、ローカル仮想メモリ107における画面メモリ109の参照座標が変化するが、そこには画像が存在しないので、ステップS19及び25の様にサーバーに矩形領域の変更が知らされ、ステップS20、26の様にリモート実メモリ108の所定の位置からビットマップ情報がクライアントに転送される。その後サーバーもクライアントもそれぞれ次のイベント待ちに入る。

【0160】以上説明したように、本実施例によれば、アプリケーションからオペレーティングシステムに対して発行される描画関数コールを捉えて前記サーバーに前記描画関数コールに対応するスクリプトを発行し、該スクリプトを解釈して前記サーバー上に確保された画像描画メモリ上に画像を描画し、該描画された描画画像デー

タを前記ネットワークを介して前記画像メモリから前記オペレーティングシステムに取り込み、画面メモリに描画するので、クライアントで処理可能なデータ量を越える大容量の画像描画処理を高速に行うことができる。

【0161】また、本実施例によれば、アプリケーションからオペレーティングシステムに対して発行される描画関数コールを捉えて前記サーバーに前記描画関数コールに対応するスクリプトを発行し、該スクリプトを解釈して前記サーバー上に確保された画像描画メモリ上に画像を描画し、該描画された描画画像データを前記ネットワークを介して前記画像メモリからローカル仮想メモリを介して前記オペレーティングシステムに取り込み、画面メモリに描画するので、クライアントで処理可能なデータ量を越える大容量の画像描画、特に編集画面に対応した描画処理を高速に行うことができる。

【0162】また、本実施例によれば、各クライアントから画像描画開始を指示するスクリプトを前記サーバーに発行し、該発行されたスクリプトに基づいて画像描画プログラムを起動するとともに、画像描画メモリを確保し、該確保された画像メモリに対応するパラメータを前記クライアントに返信して、前記クライアントからネットワークを介して受信する各描画命令に基づいて前記画像メモリに所望の画像を描画するので、クライアントの画像処理能力を越える画像編集処理を行うことができる。

【0163】また、本実施例によれば、サーバーは、各クライアントから画像描画開始を指示するスクリプトを解析して、クライアントの出力デバイスに対応する解像度よりも高解像度の画像描画を保証する画像描画メモリを確保するので、クライアントの画像処理能力を越える高解像度の画像編集処理を高速に行うことができる。

#### 【0164】

【発明の効果】以上説明した様に本発明によれば、サーバーに描画命令を出力し、出力した前記描画命令を解釈してサーバー上に確保された画像描画メモリ上に描画された画像データを取得し、取得した画像データを画面メモリに表示することで、クライアントでの画像描画処理をサーバーに代行させることができるという効果を奏する。

【0165】以上説明した様に本発明によれば、ホストコンピュータから描画命令を入力し、入力した前記描画命令を解釈して、確保した画像描画メモリ上に画像データを描画し、描画した画像データをホストコンピュータの画面メモリに表示する為に、画像データをホストコンピュータへ出力することで、クライアントでの画像描画処理を代行し画像編集処理できるという効果を奏する。

#### 【図面の簡単な説明】

【図1】本発明の一実施例を示すマルチメディアサーバ

一のシステム構成を説明するブロック図である。

【図2】図1に示したセンタサーバーを介した既存サーバーのアクセス方法を説明する図である。

【図3】本発明に係るマルチメディアサーバーにおけるメッセージの構造を説明する図である。

【図4】本発明に係るマルチメディアサーバーにおける第1のプロセススクリプトの送出手順を説明するブロック図である。

【図5】本発明に係るリアルタイム処理を説明する図である。

【図6】本発明に係る他のリアルタイム処理を説明する図である。

【図7】本発明に係るバッチ処理を説明する図である。

【図8】本発明に係るプロセススクリプトを用いた通信方法を説明する図である。

【図9】本発明に係るマルチメディアサーバーにおける第2のプロセススクリプトの送出手順を説明するブロック図である。

【図10】本発明に係るクライアントからセンタサーバーへ転送されるスクリプト例を示す図である。

【図11】本発明に係るクライアント、アプリケーションとセンタサーバーと通信プロトコルの一例を示す図である。

【図12】本発明におけるセンタサーバーと交換機を結合したマルチメディアサーバーとによるシステム構成を説明するブロック図である。

【図13】本発明に係るマルチメディアサーバーを画像描画アクセラレータとして機能させる場合の構成を説明するブロック図である。

【図14】本発明に係る画像描画アクセラレータのシステムブロック図である。

【図15】本発明に係るネットワーク描画アクセラレータの具体的なブロック図である。

【図16】本発明に係るマルチメディアサーバーにおけるローカル仮想メモリ処理の概要を示すチャートである。

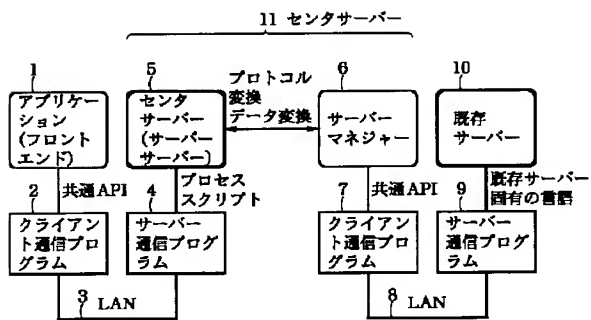
【図17】本発明に係る画像描画アクセラレータの動作を示すフローチャートである。

#### 【符号の説明】

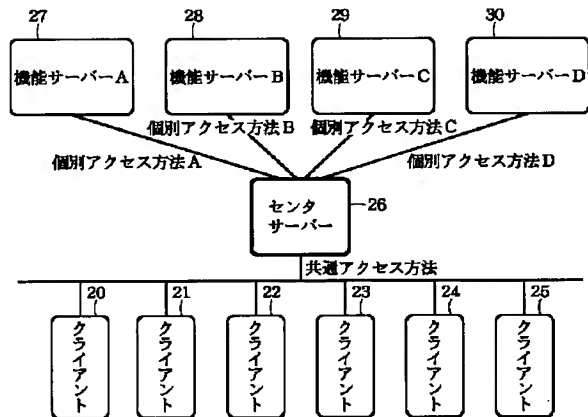
201 CPU  
202 ROM  
203 RAM  
204 LAN/I/F  
205 内部バス  
206 Bus I/O  
207 INT  
208 I/O  
209 外部バス



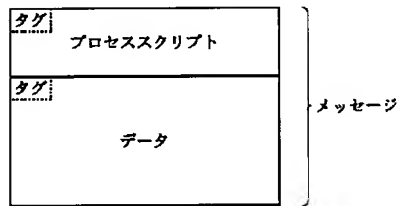
【図 1】



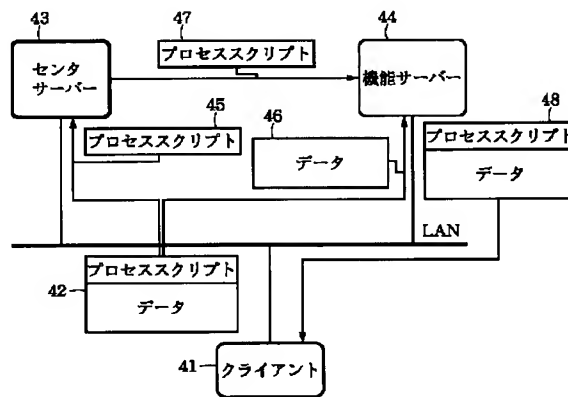
【図 2】



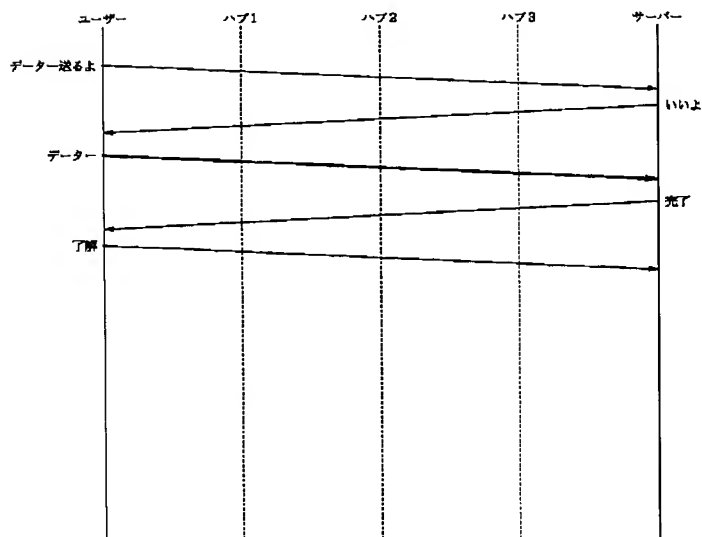
【図 3】



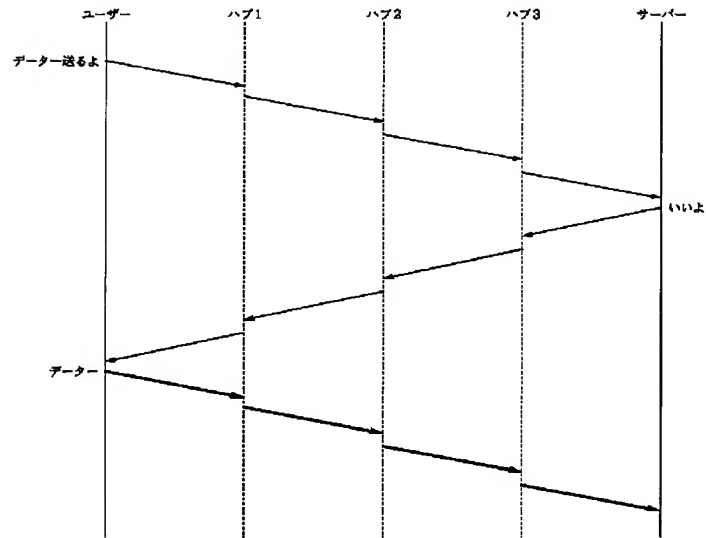
【図 4】



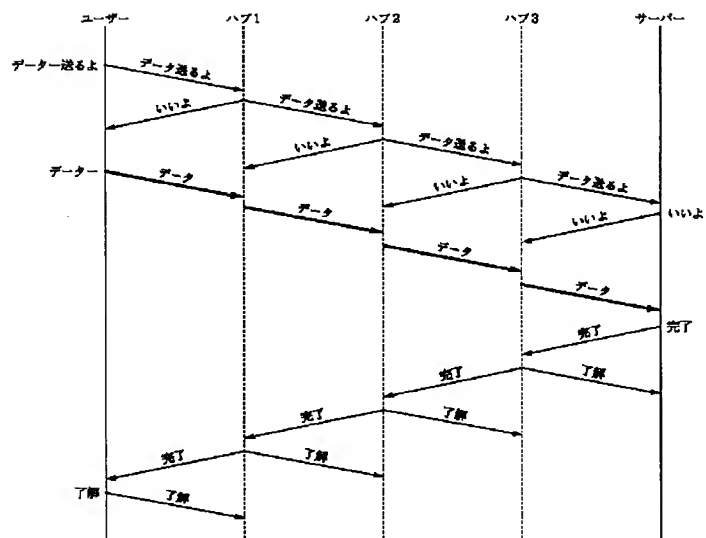
【図 5】



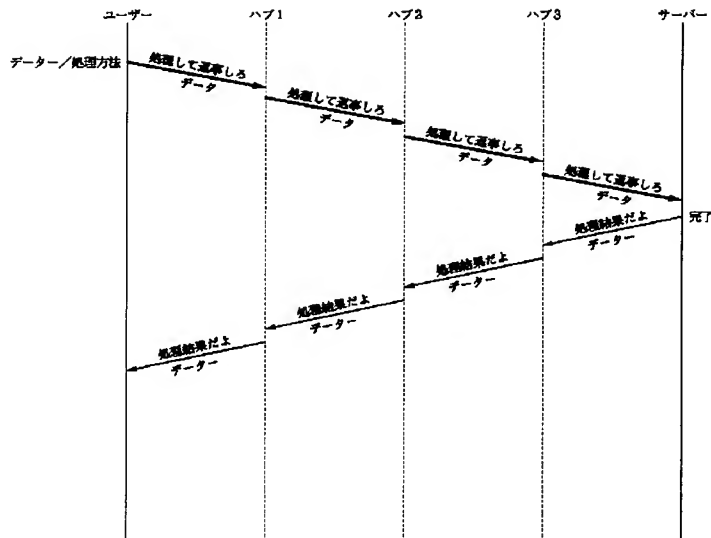
【図6】



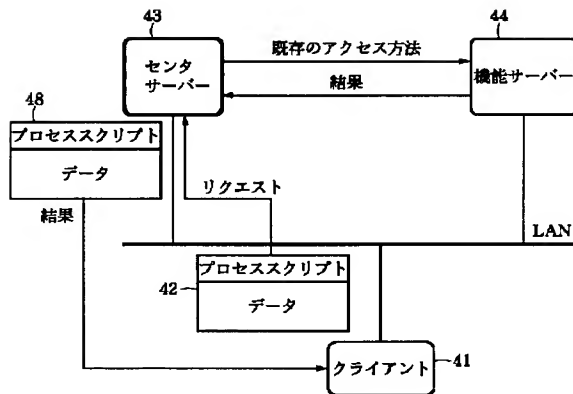
【図7】



【図 8】



【図 9】



【図 10】

(a)

```

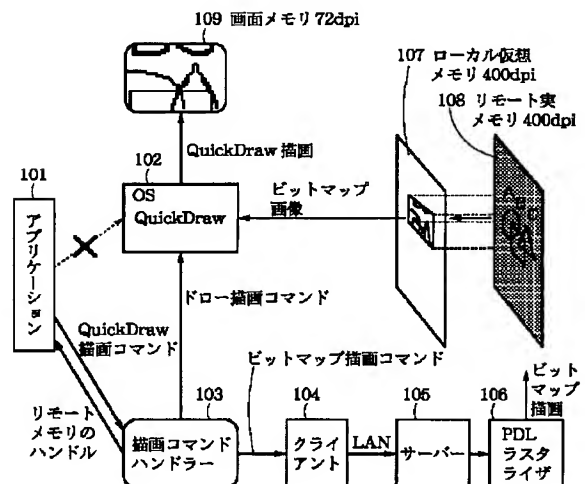
MM_open ("server", "fax_service");
MM_send (service_id, "MM_send", "G3", "data", "destination");
MM_data (service_id, "MM_send", "data_id", "length", "content_buffer");
MM_close (service_id);
  
```

(b)

```

"MM_open server fax_service"
"service_id MM_send G3 data destination"
"service_id MM_send data id length content"
"service_id MM_close"
  
```

【図 13】



【図 11】

```

アプリ クライアント          センタサーバー

MM_open ("server","fax_service");
  Message ("MM_open server fax_service")---->
  <--- Message ("service_id MM_
  open_ack")

  gservice_id = service_id;
  return (MM_open_ack);

MM_send (service_id,"MM_send"," G3"," data"," destination");
  Message ("service_id MM_send G3 data destination")-->
  <--- Message ("service_id MM_
  send_ack")

  return (MM_send_ack);

MM_data (service_id,"MM_send"," data_id"," length"," content_
  buffer");
  Message ("service_id MM_send data_id length
  content") ---->
  <--- Message ("service_id MM_
  data_ack data_id")

  gdata_id = data_id;
  return (MM_data_ack);

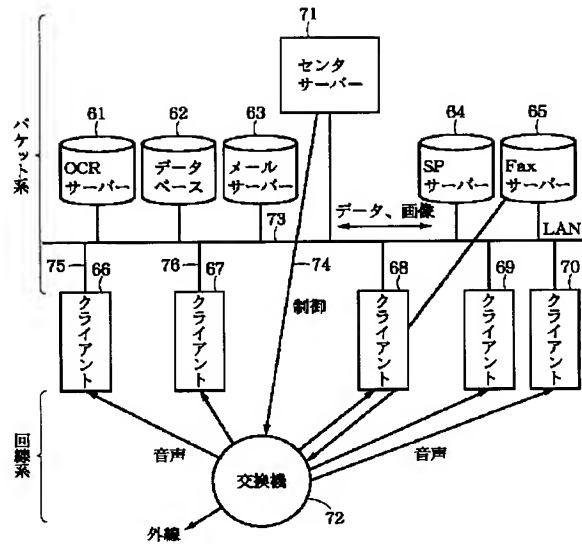
:
:

MM_close (service_id);
  Message ("service_id MM_close")---->
  <--- Message ("service_id MM_
  close_ack")

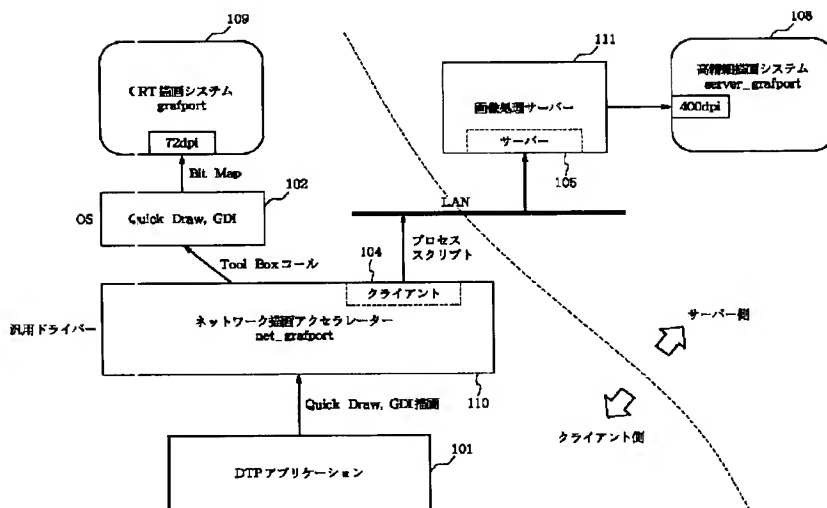
  return (MM_close_ack);

```

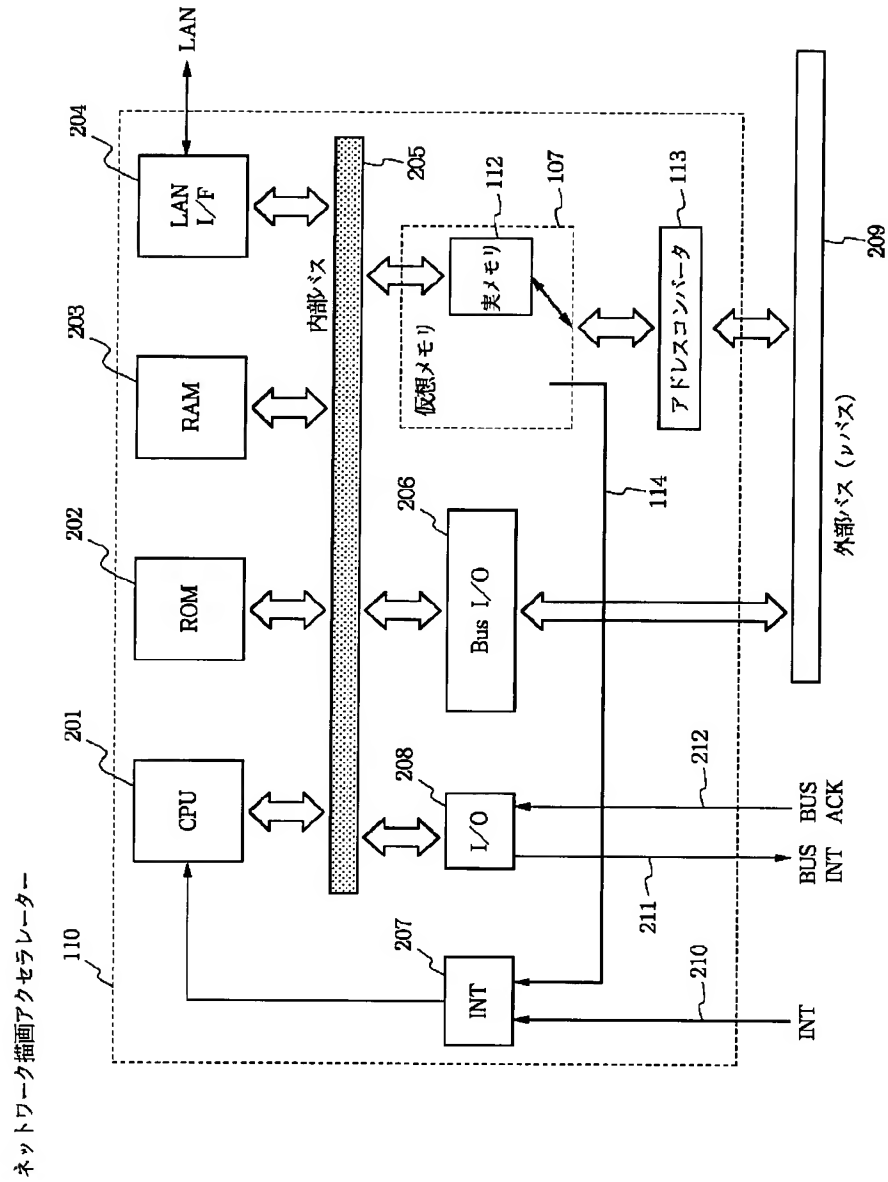
【図 12】



【図 14】



【図15】



[illegible]

【図 17】

